Amendment to the Claims:

The claims under examination in this application, including their current status and changes made in this paper, are respectfully presented.

1 (currently amended). A method for verifying that two programs are equivalent, the method comprising the steps of:

executing a first program at source hardware operating according to a first instruction set architecture;

collecting a first set of events that are created by the first program while it is being executed;

executing a second program at target hardware operating according to a second instruction set architecture;

collecting a second set of events that are created by the second program while it is being executed;

determining if the first set of events is equivalent to the second set of events by reconciling the first set of events and the second set of events; and

indicating the first program is not equivalent to the second program if an unreconciled event is discovered during the step of determining.

2 (original). The method of Claim 1, wherein the step of determining comprises the steps of:

matching each event of the first set of events to a corresponding event in the second set of events;

declaring the existence of an unreconciled event if an event in the first set of events does not have a corresponding event in the second set of events; and

declaring the existence of an unreconciled event if an event in the second set of events does not have a corresponding event in the first set of events.

3 (original). The method of Claim 2, wherein:

the step of collecting a first set of events comprises developing a logical state for each event of the first set of events;

the step of collecting a second set of events comprises developing a logical state for each event of the second set of events; and

the step of matching comprises comparing the logical state of each event of the first set of events to the logical state of the corresponding event in the second set of events.

4 (currently amended). ~~The~~ A method of ~~Claim 1~~ for verifying that two programs are equivalent, the method comprising the steps of:

executing a first program;

collecting a first set of events that are created by the first program while it is being executed;

executing a second program;

collecting a second set of events that are created by the second program while it is being executed;

determining if the first set of events is equivalent to the second set of events by reconciling the first set of events and the second set of events; and

indicating the first program is not equivalent to the second program if an unreconciled event is discovered during the step of determining;

wherein the steps of executing a first program, collecting a first set of events, executing a second program, collecting a second set of events, and determining are performed in a iterative manner such that:

after executing the first program and collecting a first event, execution of the first program is halted;

the steps of executing the second program, collecting the second set of events, and determining are performed until an event in the second set of events is found that corresponds to the first event collected;

the steps of executing the first program, collecting a first set of events, and determining are performed until events in the first set of events are found that correspond to all events in the second set of events and at least one event remains in the first set of events; and

the steps of executing the second program, collecting a second set of events, and determining are performed until events in the second set of events are found that correspond to each event in the first set of events.

5 (original). The method of Claim 4, wherein:

the step of collecting a first event comprises developing a logical state for the first event;

the step of collecting a second set of events comprises developing a logical state for each event of the second set of events;

the step of collecting a first set of events comprises developing a logical state for each event of the first set of events; and

the step of determining comprises comparing the logical state of each event of the first set of events to the logical state of the corresponding event in the second set of events or comparing the logical state of each event of the second set of events to the logical state of the corresponding event in the first set of events.

6 (currently amended). The method of Claim 1, wherein the step of indicating comprises:

activating ~~the~~ a debugging capability in ~~the~~ a software development system during the step of executing a first program, at the point where the unreconciled event is discovered; and

activating the debugging capability in the software development system during the step of executing a second program, at the point where the unreconciled event is discovered.

7 (currently amended). The method of Claim 4, wherein the step of indicating comprises:

activating ~~the~~ a debugging capability in ~~the~~ a software development system during the step of executing a first program, at the point where the unreconciled event is discovered; and

activating the debugging capability in the software development system during the step of executing a second program, at the point where the unreconciled event is discovered.

8 (original). The method of Claim 1, wherein:

the steps of collecting a first set of events and collecting a second set of events comprise treating references to address registers in instructions as symbols; and

the step of determining comprises computing the effective addresses of the analogous references in the first set of event and the second set of events when reconciling the first set of events and the second set of events.

9 (original). The method of Claim 4, wherein:

the steps of collecting a first set of events and collecting a second set of events comprise treating references to address registers in instructions as symbols; and

the step of determining comprises computing the effective addresses of the analogous references in the first set of event and the second set of events when reconciling the first set of events and the second set of events.

10 (currently amended). ~~The~~ A method of ~~Claim 1~~ for verifying that two programs are equivalent, the method comprising the steps of:

executing a first program;

collecting a first set of events that are created by the first program while it is being executed;

executing a second program;

collecting a second set of events that are created by the second program while it is being executed;

determining if the first set of events is equivalent to the second set of events by reconciling the first set of events and the second set of events; and

indicating the first program is not equivalent to the second program if an unreconciled event is discovered during the step of determining;

wherein the steps of collecting a first set of events and collecting a second set of events comprise:

determining that a first instruction is in the instruction pipeline;

calculating the current effective address delay of the instruction in the pipeline;

finding that a valid effective address for the instruction is available based on the current effective address delay of the instruction;

computing the effective address of the first instruction if a valid effective address is not available; and

reporting the effective address of the instruction.

11 (original). The method of Claim 4 wherein the steps of collecting a first set of events and collecting a second set of events comprise:

determining that a first instruction is in the instruction pipeline;

calculating the current effective address delay of the instruction in the pipeline;

finding that a valid effective address for the instruction is available based on the current effective address delay of the instruction;

computing the effective address of the first instruction if a valid effective address is not available; and

reporting the effective address of the instruction.

12 (original). The method of Claim 11 wherein:

the step of calculating comprises subtracting the number of clock cycles that have occurred since the instruction entered the pipeline from the number of clock cycles required to compute the effective address of the instruction;

the step of finding comprises determining that the current effective address delay is 0; and

the step of computing is executed if the current effective address delay is less than 0.

13 (original). The method of Claim 1, wherein the means for executing, collecting, determining, and indicating is a first verification program whereby the verification program causes a first program to be executed by a first software development system, causes a second program to be executed by a second software development system, and receives information from the first software development system and the second software development system to perform the steps of collecting, determining, and indicating.

14 (original). The method of Claim 4, wherein the means for executing, collecting, determining, and indicating is a first verification program whereby the verification program causes a first program to be executed by a first software development system, causes a second program to be executed by a second software development system, and receives information from the first software development system and the second software development system to perform the steps of collecting, determining, and indicating.

15 (currently amended). A digital software verification system, comprising:

a general purpose computer;

a first microprocessor ~~having a first architecture~~ for executing application programs;

~~a memory circuit connected to the microprocessor holding a first or target application program for execution by the microprocessor;~~

first emulation hardware, coupled between the first microprocessor and the general purpose computer, for controlling the operation of the first microprocessor according to a first instruction set architecture;

a second microprocessor for executing application programs; and

second emulation hardware, coupled between the second microprocessor and the general purpose computer, for controlling the operation of the second microprocessor according to a second instruction set architecture;

wherein the ~~first or target application program was ported from another microprocessor and verified using~~ general purpose computer is programmed to perform a method for verifying that ~~the first or~~ a target application program ~~was~~ is equivalent to a ~~the original or~~ source application program, the method comprising the steps of:

causing the first emulation hardware to execute ~~executing~~ a first program at the first microprocessor;

collecting a first set of events that are performed by the first program while it is being executed;

causing the second emulation hardware to execute ~~executing~~ a second program at the second microprocessor;

collecting a second set of events that are performed by the second program while it is being executed;

determining if the first set of events is equivalent to the second set of events; and

indicating the first program is not equivalent to the second program if a mismatch is discovered during the step of determining.

16 (currently amended). ~~The~~ A digital system of claim 15, comprising:

a microprocessor having a first architecture for executing application programs;

a memory circuit connected to the microprocessor holding a first or target application program for execution by the microprocessor; and

wherein the first or target application program was ported from another microprocessor and verified using a method for verifying that the first or target application program was equivalent to the original or source application program, the method comprising the steps of:

executing a first program;

collecting a first set of events that are performed by the first program while it is being executed;

executing a second program;

collecting a second set of events that are performed by the second program while it is being executed;

determining if the first set of events is equivalent to the second set of events; and

indicating the first program is not equivalent to the second program if a mismatch is discovered during the step of determining;

wherein the steps of executing a first program, collecting a first set of events, executing a second program, collecting a second set of events, and determining are performed in a iterative manner such that:

after executing the first program and collecting a first event, execution of the first program is halted;

the steps of executing the second program, collecting the second set of events, and determining are performed until an event in the second set of events is found that corresponds to the first event collected;

the steps of executing the first program, collecting a first set of events, and determining are performed until events in the first set of events are found that correspond to all events in the second set of events and at least one event remains in the first set of events; and

the steps of executing the second program, collecting a second set of events, and determining are performed until events in the second set of events are found that correspond to each event in the first set of events.